APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/614,970	07/08/2003	Mitchell Alsup	5500-81600	8802

53806 7590 06/12/2007	**EXAMINER**	
MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL (AMD)	GEIB, BENJAMIN P	
P.O. BOX 398		
AUSTIN, TX 78767-0398	ART UNIT	PAPER NUMBER
	2181	

MAIL DATE	DELIVERY MODE
06/12/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

MAILED

Application Number: 10/614,970
Filing Date: July 08, 2003
Appellant(s): ALSUP ET AL.

JUN 1 2 2007

Technology Center 2100

Robert C. Kowert
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 12/17/2006 appealing from the Office action

mailed 07/14/2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Tran, U.S. Patent No. 5,864,689 issued 26 July 1999

Carbine et al., U.S. Patent No. 5,630,083 issued 13 May 1997

Kling, U.S. Patent Application No. 2004/0049657 published 11 March 2004

Harris, U.S. Patent No. 6,260,138 issued 10 July 2001

"Intel Architecture Software Developer's Manual Volume 1: Basic

Architecture" published 1999

Rotenberg et al. "Trace Cache: a Low Latency Approach to High Bandwidth

Instruction Fetch" published 1996

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that

form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1, 15-17, 27-29, 39, and 41 are rejected under 35 U.S.C. 102(b) as being

anticipated by Tran, U.S. Patent No. 5,864,689. In addition, "Intel Architecture Software

Developer's Manual – Volume 1: Basic Architecture" (Herein referred to as Manual) is

cited as extrinsic evidence for showing the operation of x86 CALL and RETURN

instructions.

Referring to claim 1, Tran has taught a microprocessor, comprising:

a dispatch unit *(microcode unit and instruction decode unit; Fig. 1, components*

45 and 36) configured to dispatch operations *(column 4, lines 29-41; column 6, lines 17-*

46);

a scheduler *(reservation station)* coupled to the dispatch unit and configured to

schedule dispatched operations for execution *(column 6, lines 51-63)*;

wherein in response to receiving a microcoded instruction *(x86 CALL instruction)*,

the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call

operation that includes a tag *(target address)* identifying a microcode subroutine

associated with the microcoded instruction *(column 8, line 55 – column 9, line 4) [All*

instruction decoded by the instruction decode unit (including an x86 CALL instruction)

are dispatched to the execution units (i.e. execute units and/or load/store unit) (Tran;

column 6, lines 24-28). *As the execution units includes reservation stations (column 6,*

lines 51-63), the instructions are dispatched from the instruction decode unit (part of the

dispatch unit) to a reservation station (i.e. scheduler)].

Referring to claims 15, 27, and 39, taking claim 15 as exemplary, <u>Tran</u> has

taught the microprocessor of claim 1, wherein a same opcode is used to specify the

microcode subroutine call operation and a non-microcode subroutine call operation

(column 4, lines 42-54).

Referring to claims 16 and 28, taking claim 16 as exemplary, <u>Tran</u> has taught the

microprocessor of claim 1, wherein the microcode subroutine includes a return

operation, wherein the return operation pops a return address from the stack, wherein

execution of the microcode subroutine call operation pushes the return address onto the stack *(column 4, lines 42-54)*.

Referring to claim 17, <u>Tran</u> has taught a computer system, comprising:

A system memory *(main memory; Fig. 16, component 16)*; and

A microprocessor *(microprocessor; Fig. 5, component 12)* coupled to the system memory; wherein the microprocessor comprises:

a dispatch unit *(microcode unit and instruction decode unit; Fig. 1, components 45 and 36)* configured to dispatch operations *(column 4, lines 29-41; column 6, lines 17-46)*;

a scheduler *(reservation station)* coupled to the dispatch unit and configured to schedule dispatched operations for execution *(column 6, lines 51-63)*;

wherein in response to receiving a microcoded instruction *(x86 CALL instruction)*, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag *(target address)* identifying a microcode subroutine associated with the microcoded instruction *(column 8, line 55 – column 9, line 4) [All instruction decoded by the instruction decode unit (including an x86 CALL instruction) are dispatched to the execution units (i.e. execute units and/or load/store unit) (Tran; column 6, lines 24-28). As the execution units includes reservation stations (column 6, lines 51-63), the instructions are dispatched from the instruction decode unit (part of the dispatch unit) to a reservation station (i.e. scheduler)]*.

Referring to claim 29, Tran has taught a method comprising:

receiving a stream of instructions *(column 6, lines 13-16)*;

detecting a microcoded instruction within the stream of instructions, wherein the microcoded instruction immediately precedes an other instruction in program order *(column 4, lines 36-41)*;

in response to said detecting, dispatching a microcode subroutine call operation that identifies a microcode subroutine associated with the microcoded instruction *(column 8, line 55 – column 9, line 4)* , wherein the microcode subroutine call operation pushes an address of the other instruction onto a stack *(x86 Call instruction - column 4, lines 42-54)*; and

executing a plurality of operations included in the microcode subroutine, wherein the plurality of operations include a return operation, and wherein execution of the return operation pops the address from the stack *(x86 Return instruction - column 4, lines 42-54)*.

Referring to claim 41, Tran has taught a system comprising:

Means for receiving a stream of instructions, decoding each non-microcoded instruction within the stream of instruction into one or more operations, and dispatching each of the one or more operations *(decode unit; column 6, lines 17-28)*;

Means for executing dispatched operations *(execution unit; column 6, lines 32-43)*;

Wherein the means for receiving the stream of instructions are configured to detect a microcoded instruction within the stream of instruction and to responsively

dispatch a microcode subroutine call operation that identifies a microcoded subroutine associated with microcoded instruction *(column 8, line 55 – column 9, line 4)*;

Wherein the means for executing dispatched operations are configured to push an address onto a stack when executing the microcode subroutine call operation (x86 Call instruction), wherein the address identifies an operation generated by decoding a non-microcoded instruction immediately subsequent to the microcoded instruction within the stream of instructions *(column 4, lines 42-54)*.

Claim 40 is rejected under 35 U.S.C. 102(b) as being anticipated by Carbine et al., U.S. Patent No. 5,630,083 (Herein Referred to as Carbine).

Referring to claim 40, Carbine has taught a method comprising:

Dispatching one or more operations included in a first microcode subroutine and one or more operations included in a second microcode subroutine *(executing a first and second generic microcode routine; column 12, lines 36-56)*, wherein said dispatching the one or more operations in the first microcode subroutine comprises performing register name replacements using replacement register names stored in a first alias table element *(micro-alias register)* and wherein said dispatching the one or more operation in the second microcode subroutine comprises performing register name replacements using replacement register names stored in a second alias table element *(micro-alias register) (Each generic microcode routine has micro-alias registers that it uses to replace register names within the routine; column 12, lines 36-56)*;

Subsequent to said dispatching, detecting a branch misprediction within the first

microcode subroutine *(a mispredicted micro-branch; column 23, lines 12-35)*;

In response to said detecting, replacing register names within one or more other

operations included in the first microcode subroutine with replacement register names

stored in the first alias table element *(micro-alias register) (When operation flow restarts*

at the actual micro-branch target, the register names will be replaced with that stored in

the micro-alias register; column 23, lines 12-35; column 12, lines 36-56; See Fig. 14,

component 1430); and

Dispatching the one more other operations subsequent to said replacing (See

Fig. 14, component 1440).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set
> forth in section 102 of this title, if the differences between the subject matter sought to be patented and
> the prior art are such that the subject matter as a whole would have been obvious at the time the
> invention was made to a person having ordinary skill in the art to which said subject matter pertains.
> Patentability shall not be negatived by the manner in which the invention was made.

Claims 2-6, 18-22, and 30-34 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Tran in view of Carbine et al., U.S. Patent No. 5,630,083 (Herein

referred to as Carbine).

Referring to claims 2, 18, and 30, taking claim 2 as exemplary, Tran has taught

the microprocessor of claim 1.

Tran has not explicitly taught that the dispatch unit is further configured to

dispatch an operation that provides one or more register names for use as replacement

register names within the microcode subroutine.

Carbine has taught a dispatch unit *(MS unit; Fig. 5, component 534)* that is

configured to dispatch an operation *(LOADUAR signal)* that provides one or more

register names for use as replacement register names within the microcode subroutine

[Carbine; column 12, lines 24-67].

At the time the invention was made, it would have been obvious to a person of

ordinary skill in the art to modify the dispatch unit of Tran to dispatch an operation that

provides one or more register names for use as replacement register names with the

microcode subroutine as taught by Carbine.

The suggestion/motivation for doing so would have been that a generic

microcode routine can be used by any number of other microcode programs *(Carbine;*

column 12, lines 49-52).

Referring to claims 3, 19, and 31, taking claim 3 as exemplary, Tran and Carbine

have taught the microprocessor of claim 2, wherein the dispatch unit is configured to

allocate an alias table element *(Carbine; micro-alias register; Fig. 5, component 562)* to

store the one or more register names in response to handling the operation *(Carbine;*

column 12, lines 15-35).

Referring to claims 4 and 20, Tran and Carbine have taught the microprocessor

of claim 2, wherein the dispatch unit is configured to maintain multiple allocated alias

table elements *(Carbine; micro-alias registers; Fig. 5, component 562)* at a same time *(Carbine; column 12, lines 15-35)*.

Referring to claims 5, 21, 33, and 34, taking claim 5 as exemplary, <u>Tran</u> and <u>Carbine</u> have taught the microprocessor of claim 4, wherein each of the multiple allocated alias table elements (micro-alias registers) is associated with a respective microcode subroutine *(Carbine; The generic microcode routine that uses that particular allocated micro-alias register; See column 12, lines 49-56)*, wherein the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved *[Carbine; If a micro-branch (a branch in microcode) is mispredicted the dispatch unit updates/maintains the micro-alias registers so that execution can restart at the actual target of the micro-branch. Therefore, the dispatch unit maintains each micro-alias register until all branch operations within the respective microcode subroutine have resolved; See column 23, lines 12-35]*.

Referring to claims 6 and 22, taking claim 6 as exemplary, <u>Tran</u> and <u>Carbine</u> have taught the microprocessor of claim 4, wherein in response to detection of a branch misprediction with a microcode subroutine, the dispatch unit is configured to perform replacements within one or more microcode operations included within the microcode subroutine according to the one or more register names stored within a respective alias table element and to dispatch the one more microcode operations subsequent to performing the replacements *(Carbine; column 23, lines 12-35)*.

Claims 7-8, 23-24, and 35-36 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Tran and Carbine as applied to claim 2 above, and further in view of

Rotenberg et al., "Trace Cache: a Low Latency Approach to High Bandwidth Instruction

Fetching" (Herein referred to as Rotenberg).

Referring to claims 7, 23, and 35, taking claim 7 as exemplary, Tran and Carbine

have taught the microprocessor of claim 2 wherein the dynamic instruction stream

includes a microcode subroutine call operation (Tran; column 8, line 55 – column 9, line

4) and the one or more register names for use as replacement register names (Carbine;

part of the LOADUAR instruction - column 12, lines 24-67).

Tran and Carbine have not explicitly taught a trace cache coupled to the dispatch

unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the

trace cache entry includes instructions from the dynamic instruction stream.

Rotenberg has taught a trace cache coupled to the dispatch unit, wherein the

trace cache includes a trace cache entry (Rotenberg; See Section 1.1 and Fig. 2).

Rotenberg has further taught that a trace stored in the trace cache entry includes

instructions from the dynamic instruction stream (Rotenberg; See Section 1.1).

At the time the invention was made, it would have been obvious to a person of

ordinary skill in the art to modify the system of Tran and Carbine to include a trace

cache coupled to the dispatch unit, wherein the trace cache includes a trace cache

entry; wherein a trace stored in the trace cache entry includes instructions from the

dynamic instruction stream as taught by Rotenberg.

The suggestion/motivation for doing so would have been that the trace cache improves the performance of the microprocessor *(Rotenberg; See Abstract)*.

Referring to claims 8, 24, and 36, taking claim 8 as exemplary, Tran, Carbine, and Rotenberg have taught the microprocessor of claim 7, wherein in response to receiving the trace from the trace cache, the dispatch unit is configured to allocate an alias table *(Carbine; micro-alias register; Fig. 5, component 562)* to store the one or more register names *(In response to receiving the LOADUAR instruction, a micro-alias register is allocated - Carbine; column 12, lines 15-35)*.

Claims 9, 10, 25, and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tran in view of Kling, U.S. Patent Publication No. 2004/0049657.

Referring to claims 9, 25, and 37, taking claim 9 as exemplary, Tran has taught the microprocessor of claim 1.

Tran has not explicitly taught that the dispatch unit is configured to store the microcode subroutine in one or more microcode traces.

Kling has taught a dispatch unit *(Kling; microcode unit; Fig. 2, component 46; paragraph 13)* that is configured to store a microcode subroutine in one or more microcode traces *(Kling; paragraph 32 on page 4)*.

At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the dispatch unit of Tran to be configured to store the microcode subroutine in one or more microcode traces as taught by Kling.

The suggestion/motivation for doing so would have been that the microcode traces enable more efficient processing of the instructions included in the trace upon subsequent invocations *(Kling; paragraph 32 on page 4).*

Referring to claim 10, Tran and King have taught the microprocessor of claim 9, wherein the one or more microcode traces are stored within a memory *(Kling; trace cache; paragraph 32 on page 4).*

Claims 11-14, 26, and 38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tran and Kling as applied to claim 9 above, and further in view of Harris, U.S. Patent No. 6,260,138.

Referring to claims 11, 26, and 38, taking claim 11 as exemplary, Tran and Kling have taught the microprocessor of claim 9, wherein microcode operations are stored in one or more microcode traces *(see claim 9).*

Tran and Kling have not explicitly taught that each operation includes an associated liveness indication.

Harris has taught an instruction cache wherein each operation includes an associated liveness indication *(Harris; priority tag; column 3, lines 28-31).*

At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the trace of Tran and Kling to include an associated liveness indication *(priority tag)* with each operation as taught by Harris.

The suggestion/motivation for doing so would have been that priority tags provide "enhanced performance with respect to conventional predictive branching without the additional cost of duplicated hardware as required by multiway branching *(Harris; column 10, lines 20-24)*.

Referring to claim 12, Tran, Kling, and Harris have taught the microprocessor of claim 11, wherein the dispatch unit *(Harris; dispatch unit; Fig. 8, component 26)* is configured to determine whether each microcode operation stored in one of the one or more microcode traces is executable *(i.e. on the predicted path)* dependent on at least one of: a branch prediction and the associated liveness indication *(priority tag)* *(Harris; column 5, lines 27-34)*;

wherein the dispatch unit is configured to signal whether each microcode operation stored in the one of the one or more microcode traces is executable when dispatching that microcode operation to the scheduler *(Harris; execution buffer; Fig. 8, component 126)* [Harris; The dispatch unit signals whether each operation is executable by sending the priority tag to the scheduler (execution buffer); column 9, lines 38-39];

wherein the scheduler *(execution buffer)* is configured to store an associated indication *(priority tag)* for each dispatched microcode operation indicating whether that dispatched microcode operation is executable *(Harris; column 9, lines 38-46)*.

Referring to claim 13, <u>Tran</u>, <u>Kling</u>, and <u>Harris</u> have taught the microprocessor of

claim 12, wherein if the branch predication is incorrect, the scheduler is configured to

update the associated indication for at least one dispatched microcode operation

(<u>Harris</u>; column 8, lines 6-39).

Referring to claim 14, <u>Tran</u>, <u>Kling</u>, and <u>Harris</u> have taught the microprocessor of

claim 11, wherein the dispatch unit is configured to selectively dispatch microcode

operations included in the one or more microcode traces dependent upon at least one

of: the associated liveness indication (priority tag) and branch prediction *(<u>Harris</u>; The*

dispatch unit selects instruction for dispatch from those having the highest priority (as

indicated by the priority tag) to those having the lowest - column 7, lines 44-51).

(10) Response to Argument

First ground of rejection:

Claims 1, 15, 16, 17, 27, 28, and 39:

In response to the appellant's argument that Tran fails to teach

receiving/detecting a microcoded instruction, the examiner asserts that appellant

appears to be reading "microcoded instruction" too narrowly. Tran states that

"instruction decode unit 36 stalls upon <u>detection</u> of an <u>instruction</u> to be performed by the

microcode unit" [Tran; column 4, lines 36-41] and gives an x86 CALL instruction an

example of such an instruction [Tran; column 4, lines 52-59]. Thus, an x86 CALL

instruction is a microcoded instruction in that it is performed (at least in part) by the

microcode unit. Therefore, Tran has taught receiving/detecting a microcoded instruction

(i.e. a CALL instruction).

In response to the appellant's argument that Tran "does not describe dispatching

a microcode subroutine call operation to a scheduler", the examiner notes the flow of

instructions in general through the processor of Tran. Tran describes that "instructions

are fetched from instruction cache 32 and conveyed to instruction decode unit 36 for

decode and dispatch to an execute unit 38 or load/store unit 40" [Tran; column 6, lines

13-16]. Tran further states that "instruction decode unit 36 decodes each instruction

fetched from instruction cache 32. Instruction decode unit 36 dispatches the instruction

to execute units 38 and/or load/store unit 40" [Tran; column 6, lines 24-28]. Therefore, it

can be seen that Tran has taught that every instruction fetched is dispatched to the

execute units 38 and/or load/store unit 40. The execute units 38 and load/store unit 40

"include one or more reservation stations for storing instructions whose operands have

not yet been provided" [Tran; column 6, lines 50-52]. As noted in appellant's

specification [See page 15, line 30 – page 16, line 3], reservation stations are

schedulers. Therefore, it follows that Tran has taught that instructions fetched from the

instruction cache 32 are decoded by the decode unit 36 and dispatched to reservation

stations (i.e. schedulers) within either execute units 38 or load/store unit 40.

The examiner would like to point out the operation of the instruction decode unit

36 of Tran in order to further clarify the examiner's position regarding "microcode

subroutine call operation". Tran states that decoding "refers to transforming the

instruction from the format defined by the microprocessor architecture employed by

microprocessor 12 into a second format ... the second format comprises decoded

control signals ... in order to perform the operation the instruction defines" [Tran;

column 6, lines 17-24]. Therefore, it can be seen that an instruction is decoded into

another format that comprises the various signals that are required to perform the

operation of an instruction. When this decoding process is applied to the x86 CALL

instruction, which Tran gives as an example of a "subroutine call instruction" [Tran;

column, lines 52-54], the instruction is transformed into a second format that the

examiner refers to as the claimed "microcode subroutine call operation".

Therefore, it can be seen that the instruction decode unit 36 transforms an x86

CALL instruction into a second format (i.e. a microcode subroutine call operation) that is

dispatched to the reservation stations.

The appellant argues that since the indication of the x86 CALL instruction is sent

to the microcode unit 45, a microcode subroutine call operation is not dispatched to the

reservation stations. While the appellant is correct that an indication of x86 CALL

instruction is sent to the microcode unit 45, this indication is not the only signal that is

part of the "second format" of the x86 CALL instruction (i.e. the microcode subroutine

call operation). As noted above, all instructions or, more accurately, their "second

formats" are dispatched to the reservation stations. Although Tran is clear in describing

that all instructions are dispatched the reservation stations, the examiner has provided

extrinsic evidence ("Intel Architecture Software Developer's Manual Volume 1: Basic

Architecture" (Hereinafter "Manual") in order to show why the x86 CALL instruction in

particular would need to be dispatched. Specifically, the operation of the x86 CALL

instruction involves actions besides those performed by the microcode unit. Appellant
has argued that the execution of the x86 CALL instruction does not require that these
actions be performed by any of the specific components recited in the claims. While
Manual alone does not require that these actions be performed by the components
recited in the claims, this evidence was cited merely to give support or reasoning for
why the x86 CALL instruction would be dispatched to a reservation station. That is,
Tran states that all instructions (including the x86 CALL instruction) are dispatched to
the reservation stations and Manual gives support or reasoning as to why this would be
so in the case of an x86 CALL instruction.

Claims 29 and 41:

In response to the appellant's argument regarding detecting a microcoded
instruction, the examiner points out that, as described above, Tran has taught detecting
an x86 CALL instruction, which is a microcoded instruction.

In response to the appellant's argument that Tran has not taught "detecting a
microcoded instruction that immediately precedes another instruction in program order",
the examiner notes that one of the actions performed by an x86 CALL instruction (see
x86 CALL instruction definition on page 4-5 in Manual) is to save the current instruction
pointer (referred to in Manual as EIP) so that the executing program can properly return.
Since execution resumes at the instruction pointed to by the saved instruction pointer
(see x86 CALL instruction definition on page 4-5 in Manual), this instruction pointer

points to the next instruction in program order. Therefore, the x86 CALL instruction (i.e. a microcoded instruction) precedes another instruction in program order.

In response to the appellant's argument regarding dispatching a microcode subroutine call operation, the examiner points out that, as described above, Tran has taught dispatching the "second format" (i.e. microcode subroutine call operation) of an x86 CALL instruction.

In response to the appellant's argument that Tran has not taught that a microcode subroutine call operation pushes an address of the other instruction onto a stack, that the microcode subroutine includes a return operation, and that execution of the return operation pops the address from the stack, the examiner notes that Tran has taught executing an x86 CALL instruction, which, as noted above, involves the dispatch of a microcode subroutine call operation. This microcode subroutine call operation (i.e. the "second format" of the x86 CALL instruction) involves, as described above, saving the address of the other instruction (i.e. the current instruction pointer). As described in Manual, this saving of the address involves pushing the address onto a stack (see x86 CALL instruction definition on page 4-5 in Manual). Tran has further taught that the microcode subroutine executed as part of an x86 CALL instruction concludes with the execution of a subroutine return instruction, namely an x86 RET instruction [Tran; column 4, lines 42-54]. As described in Manual, an x86 RET instruction involves popping the instruction pointer (i.e. the address of the other instruction) from the stack (see x86 RET instruction definition on page 4-6 in Manual).

Second ground of rejection:

Claim 40:

In response to the appellant's argument that Carbine has not taught multiple or

separate micro-alias registers (i.e. alias table elements), the examiner directs attention

to column 12, line 36, of Carbine, which recites "the micro-alias **registers** 562 are

particularly useful...". Therefore, Carbine clearly has taught multiple micro-alias

registers, not just a single micro-alias register as asserted by appellant.

In response to the appellant's argument that Carbine has not taught dispatching

multiple microcode subroutines, the examiner notes that while Carbine refers to the

execution/dispatch of a rounding routine to describe a generic microcode subroutine,

Carbine indicates that there are other generic microcode subroutines

executed/dispatched. Specifically, Carbine refers to the rounding routine as only an

example of a generic microcode [Carbine; column 12, lines 39-41] thereby indicating

that there is a plurality of generic microcode subroutines being executed/dispatched.

Carbine also states that "the registers are not hard-coded into <u>any</u> routine...". [Carbine;

column 12, lines 52-56] further indicating that there is a plurality of generic microcode

subroutines being executed/dispatched.

Third ground of rejection:

Claims 2, 18, and 30:

In response to the appellant's argument that the microcode sequencing unit of

Carbine is not analogous to the dispatch unit recited in the claims, the examiner notes

that appellant is reading "dispatch unit" too narrowly. The microcode sequencing unit of

Carbine sends (i.e. dispatches) the LOADUAR signal that performs that operation of

providing "one or more register names for use as replacement register names within the

microcode subroutine" as claimed. Specifically, the LOADUAR signal loads a micro-

alias register [Carbine; column 12, lines 24-30] that holds an address of a register (i.e. a

register name) that is to be used as a replacement within a microcode subroutine

[Carbine; column 12, lines 36-49]. Therefore, the LOADUAR signal performs the

operation as claimed and the microcode sequencing unit is analogous to the claimed

dispatch unit.

In response to the appellant's argument that the LOADUAR signal is not an

operation, the examiner points out that, in general, the execution of instruction in a

microprocessor consists of nothing more than the sending/dispatching of various signals

throughout the processor. The appellant appears to be reading "operation" too

narrowly. As described above, the LOADUAR signal of Carbine performs the claimed

operation and is an operation.

Claims 3, 19, and 31:

The appellant argues that Tran in view of Carbine have not taught "wherein the

dispatch unit is configured to allocate an alias table element to store the one or more

register names in response to handling the operation" stating that the "micro-alias

register is clearly not a table in which elements are allocated". The examiner notes for

clarification that it is not the micro-alias register of Carbine that is indicated as being a

table as asserted by appellant. Instead, a micro-alias register is indicated as being an

alias table <u>element</u>. Therefore, the plurality of micro-alias registers make up an alias

table.

In response to the appellant's argument that the dispatch unit does not <u>allocate</u> a

micro-alias register (i.e. alias table element), the examiner notes that, as described

above, there are a plurality of micro-alias registers. The dispatch unit sends a

LOADUAR signal to load a micro-alias register, thereby allocating a micro-alias register.

The appellant's argues that the examiner has not provided a reason for

combining the teachings of Tran and Carbine with regards to the limitations of claim 3.

The appellant is incorrect. The limitations of claim 3 are taught as a result of the

combination of Tran and Carbine as described the rejection of claim 2. Motivation for

combining Tran and Carbine is given in the rejection of claim 2.

Claims 4, 5, 20, 21, 33, and 34:

In response to the appellant's argument that Tran in view of Carbine have not

taught that the dispatch unit is configured to maintain multiple allocated alias table

elements at a same time, the examiner notes that, as described above regarding claim

40, Carbine has taught maintaining a plurality of micro-alias registers (i.e. alias table

elements).

The appellant argues that Tran in view of Carbine have not taught that "each of

the multiple allocated alias table elements is associated with a respective microcode

subroutine". As described above regarding claim 40, Carbine has taught maintaining a

plurality of micro-alias registers. As further described above regarding claim 40, Carbine has taught the micro-alias registers are used for replacement register names for a generic microcode subroutine. Therefore, each micro-alias register is associate with the generic microcode subroutine for which it is holding a replacement register name.

In response to the appellant's argument that Tran in view of Carbine have not taught that "the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved", the examiner notes that the appellant is reading the term "maintain" too narrowly. The appellant asserts the term "update" is the opposite of maintaining and describes what the term "maintain" is intended to mean. The examiner notes that although the claims are interpreted in light of the specification, absent an explicit and deliberate definition of a term, the term is subject to its broadest reasonable interpretation. Clearly, updating is a type of maintenance.

The appellant's argues that the examiner has not provided a reason for combining the teachings of Tran and Carbine with regards to the limitations of claims 4 and 5. The appellant is incorrect. The limitations of claim 4 and 5 are taught as a result of the combination of Tran and Carbine as described the rejection of claim 2. Motivation for combining Tran and Carbine is given in the rejection of claim 2.

Claims 6, 22, and 32:

In response to the appellant's arguments that Tran in view of Carbine have not

taught that "in response to detection of a branch misprediction within a microcode

subroutine, the dispatch unit is configured to perform replacements within one or more

microcode operations included within the microcode subroutine according to the one or

more register names stored within a respective alias table element and to dispatch the

one or more microcode operations subsequent to performing the replacements", the

examiner notes that the appellant appears to read an additional limitation into the claim.

In particular, the appellant appears to read into claim 6 a further limitation that would

require that the micro-alias registers (i.e. alias table elements) do not change values

subsequent to mispredicted branch.   There is nothing in the language of claim 6 that

requires this further limitation. The language of claim 6 merely describes the usage of

the alias table elements subsequent to a mispredicted branch. The citation of Carbine

given by the examiner (Carbine; column 23, lines 12-35) describes this usage of the

micro-alias registers subsequent to a mispredicted branch.

Fourth ground of rejection:

Claims 7, 23, and 35:

In response to appellant's argument that Tran in view of Carbine and Rotenberg

have not taught "a trace cache coupled to the dispatch unit, wherein the trace cache

includes a trace cache entry; wherein a trace stored in the trace cache entry includes

the microcode subroutine call operation and the one or more register names for use a

replacement register names", the examiner notes that he has not incorrectly quoted

claim 7 as indicated by the appellant. The appellant argues that Rotenberg has not

taught that a trace cache entry includes <u>the microcode subroutine call operation and the</u>

<u>one or more register names for use as replacement register names</u>. However, the

examiner relied upon Tran in view of Carbine to teach the limitation, not Rotenberg.

The examiner relies upon Rotenberg to teach "a trace cache coupled to the dispatch

unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the

trace cache entry includes instructions from the dynamic instruction stream". The

dynamic instruction stream, as taught by Tran in view of Carbine, includes the

microcode subroutine call operation and the one or more register names for use as

replacement register names.

Claim 8, 24, and 36:

The appellant argues that Tran in view of Carbine and Rotenberg have not taught

that "in response to receiving the trace from the trace cache, the dispatch unit is

configured to allocate an alias table to store the one or more register names" stating

that "LOADUAR is not an instruction at all". As described above regarding claim 2, the

LOADUAR signal is an operation, or instruction, that loads a micro-alias register (i.e.

alias table element) with a register name. As also described above regarding claim 2,

the LOADUAR signal is sent from the dispatch unit. Therefore, the dispatch unit

allocates/loads micro-alias registers to store one or more register names via the

LOADUAR instruction.

Fifth ground of rejection:

Claims 9, 10, 25, and 37:

The appellant argues that Tran in view of Kling have not taught that "the microcode subroutine is stored as one or more microcode traces". The appellant asserts that component 46 of Kling is not a microcode unit, but instead microcode. Kling states that "the processor 12 includes microcode 46 that..." Clearly, microcode 46 is a unit of processor 12 of Kling. The appellant acknowledges that Kling has taught a microcode trace, but asserts that Kling has not taught that a microcode subroutine is stored in one or more such traces. The examiner notes that, as indicated by the name "microcode trace", the trace comprises microcode instructions. A subroutine is merely a sequence of instructions. Therefore, the microcode inherently includes at least one microcode subroutine.

Sixth ground of rejection:

Claims 11, 26, and 38 and 12, 13, and 14:

In response to appellant's argument that Tran in view of Kling and Harris have not taught that "each microcode operation stored in the one or more microcode traces includes an associated liveness indication", the examiner notes that the appellant appears to be reading "liveness indication" too narrowly. Kling indicates that a priority tag (also referred to as a prediction bit) indicates "whether an instruction relates to a predicted path or a non-predicted path following a branch instruction". [Kling; column 4, lines 50-53] Therefore, the priority tag indicates whether the instruction belongs to the

group of instructions on the predicted or non-predicted path (i.e. which liveness group) and is a liveness indication.

Appellant further argues that Harris has not taught storing the priority tags in a trace. However, the system of Tran and Kling already stores microcode instructions within a trace. Harris is relied upon for teaching storing the priority along with the associated instruction. As microcode instructions in the system of Tran and Kling are stored within a trace, the combined system of Tran, Kling, and Harris stores priority tags along with the microcode instructions in a trace.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,
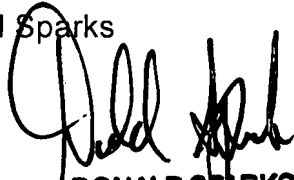
Benjamin Geib

Conferees:

Donald Sparks

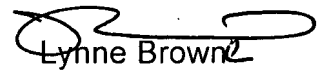**DONALD SPARKS**
SUPERVISORY PATENT EXAMINER

Lynne Brown

APPEAL PRACTICE SPECIALIST, TQA
TECHNOLOGY CENTER 2100